User Interface Design (UID) Section

User Interface Design (UID) Process

Overview

Now that we've looked at the basic principles behind HCI and the importance of involving the user in the design process (UCD), we will take a look at a four-phase interface design process that incorporates all of these concepts - the *User Interface Design Process or UID.*

While developing the interface may not be separate from the rest of the software development process, its focus is different from the other areas of software development. The focus is on the interface elements and objects that users perceive and use, rather than program functionality.

Iterative Nature of UID

Any successful design process must be iterative. You can't create a well designed interface unless you periodically go back and check with your users. Apple, IBM, Microsoft and Sun all promote an iterative design process in their user interface design guides. They show a circular process that keeps cycling through different phases.

Traditional design processes often follow a "waterfall" process that follows a linear rather than an iterative approach. This type of approach can lead to oversights that can have a tremendous impact on the cost of the development project.

Let's set development costs that occur in the analysis phase as a factor of 1. Changes made during design increase cost by a factor of 10. Changes made later in development increase that factor to 100.
The Development Methodology Rule of 10

Developers must continually incorporate iterative design revisions until they have met or exceeded the defined interface goals, or until a critical development deadline has been reached. The interface designer must always be an advocate for the user community during the design process.

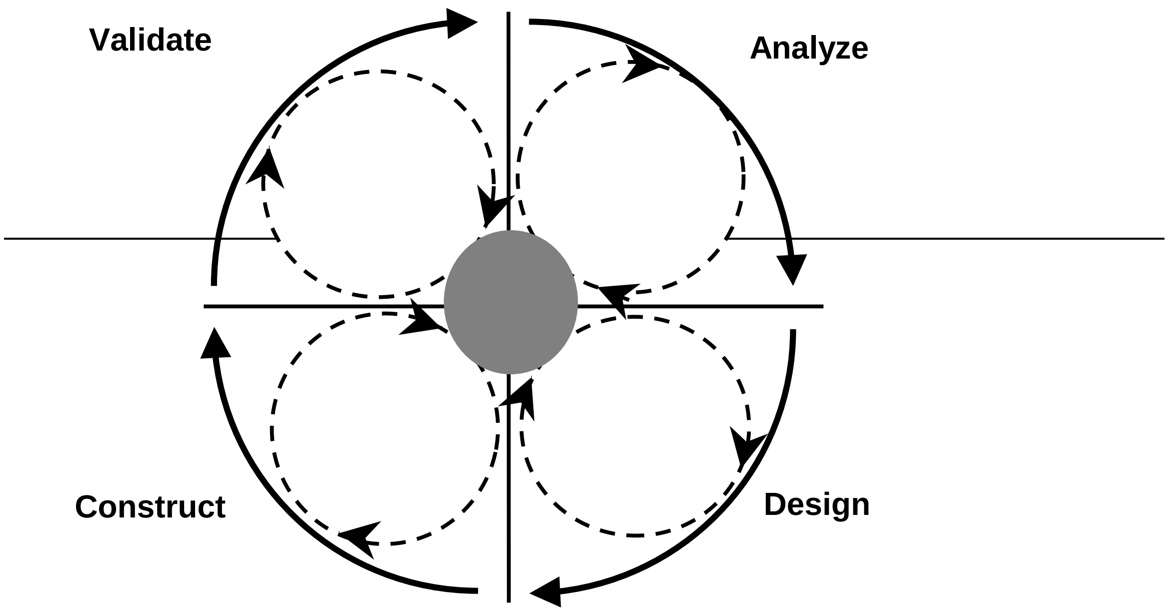
Four-phased Approach

The four phases in the UID process are:

Phase 1: *Analyze* user information.
Phase 2: *Design* the user interface.
Phase 3: *Construct* the user interface.
Phase 4: *Validate* the user interface.

In *The Elements of User Interface Design*, Theo Mandel illustrates these phases and their iterative nature in a circle to highlight the process flowing through the four major phases with spirals to show that at any time, in any phase, you may return to a central core of information, analyses, and techniques to further refine your project interface design. This illustration is shown in the figure titled *Mandel's iterative design phases*.

Mandel's iterative design phases



Phase 1: Analyze User Information
You must first start with your users. It's very important to *"Know Thy User"*. In this phase, before you design and build your application, learn about your user's capabilities and how they perform their tasks on the job. Define the tasks that the users perform to do their jobs.. Look for any hardware or software constraints in the user's current system. Your solution must satisfy both current needs and future needs, so be careful not to let current systems limit your designs to what user's can only currently do.

Here are common activities that can be performed to assist you in gathering and analyzing user information.

- **Determine User Profiles**
  Users can be broken into multiple user segments. Each segment must have a profile. User profiles answer questions about the user demographics, psychographics, skills, knowledge, and background. Conduct interviews, surveys, and focus groups, observe

and videotape users, and review other information, such as industry reports, reviews, and press or marketing materials.

- **Perform User Task Analysis**
  Determine *what* users want to do and *how* they accomplish their tasks. Some important *what* and *how* questions are:
  - What tasks do users perform?
  - What steps are taken to perform tasks?
  - What tasks are most critical?
  - What information is needed to perform each task?
  - What tools are currently used to complete each task?
  - What output is generated from user tasks?
  - How do users interact with others in the tasks?
  - How do tasks flow in the business process?

- **Gather User Requirements**
  Ask questions that answer the question, "What do users expect the application to do for them?" These requirements can be gathered in focus groups, interviews, and user surveys. Some important questions to ask are:
  - What technologies do users require?
  - How much are users and managers willing to pay for a product?
  - How much or how little training do users expect to be able to use new products?
  - Who installs products?
  - Who supports the products once they are installed?

- **Analyze User Environments**
  Ask questions that answer the question, "Where do users perform their tasks?" You must determine conditions that exist in the working environment that impact how users do their work. Make sure to collect information that includes:
  - Physical work environment - lighting, noise, space, temperature, telephones, people.
  - User location - work at home, on site, mobile
  - Human factors - vision, hearing, keyboard abilities, sitting versus standing.

- **Match Requirements to Tasks**
  Reviewing user tasks and requirements is a reality check to make sure the requirements are in line with the tasks the users are trying to perform. Also, check to see if you are going beyond what the users require to get their job done. If users really only need text information for data entry, don't give them multimedia and controls on the interface that are beyond their needs and requirements.

Phase 2: Design the User Interface

It is very tempting for the developer to start coding the application, rather than to design the interface. Studies have shown that 70 percent of development costs are in the design of the user interface. With so much of the application's success tied up in the design, it's important to follow the steps in the design process before you code.

The steps involved in the design phase are best done in sequence. These steps are:

1. **Define Usability Goals and Objectives**
   You won't be able to determine if the final product works if you haven't defined what is usable. Design goals are best stated in terms of the user's behavior and performance in accomplishing their tasks, like how long tasks take and how many errors are expected. The table titled *User Goals for Productivity* lists some examples of user goals for personal productivity or work. Determine the users' goals and design the interface to enable them to achieve their goals.

User Goals for Productivity

- Increased productivity
- Greater accuracy
- Higher satisfaction
- Ease of learning
- Ease of use

2. **Develop User Scenarios and Tasks**
   A *scenario* is a high level explanation of what the user does. A scenario is made up of a *sequence of tasks*. Tasks can further be broken down into *subtasks*. Develop as many user scenarios as possible. If you have users with a broad range of skills, make sure to develop scenarios that cover the entire skill range. For more information on developing user scenarios, read John Carroll's book, *Scenario-Based Design: Envisioning Work and Technology in System Development*.

3. **Define Interface Objects and Actions**
   In user's everyday tasks there are familiar objects that they use. The designer must understand these objects. This is probably the most difficult of all the design tasks. It requires that you define your initial set of user objects and actions based on information you collected in Phase 1 and from the scenarios you created in the previous step.

   Objects are *nouns*. Go through the scenarios and tasks and underline all of the nouns in your scenarios. *Actions* are *verbs*. Again go through the scenarios and tasks and circle all of the verbs in your scenarios You now have a first pass at identifying those objects that can have actions performed on them by a user.

   Theo Mandel's book, *The Elements of User Interface Design*, does an excellent job at describing in detail the necessary steps to accomplish this difficult task. Make sure that you review your results with your users. Don't wait until design is complete before you validate the objects and actions with the user.

4. **Determine Object Icons**
   Developers do not view objects in the same way that users do. Graphic designers should work with users to design icons that match the final objects that were identified in the previous step.

5. **Design Windows and Menus**
   When first designing the window layouts, prototyping is a very effective tool. A *lo-fidelity (lo-fi) prototype* is a paper-based prototype focusing on window information content. It can be quickly and inexpensively created, and can be easily changed. The primitive look of the paper prototype forces users and designers to think conceptually about the application flow and general layout, rather than on details of the functionality. To enhance that benefit, details of the interface navigation should be avoided.

Phase 3: Construct the User Interface

Once an initial design has been approved, you should once again *prototype*, rather than construct the interface. These more advanced methods of prototyping, called *hi-fidelity (hi-fi) prototyping*, involve the use of GUI development tools instead of paper.

It is very important to be willing to throw away a prototype. The purpose of prototyping is to quickly and easily visualize the design, not to build code that must be part of the final product. By putting off coding efforts until you have a better feel for the interface, you will avoid rework.

Phase 4: Validate the User Interface

A key piece of the iterative design process is the *usability test*. Good interface design alone does not guarantee a usable product, and usability testing cannot be substituted for a good design - both go hand in hand throughout the UID process.

The ISO defines *usability* as the effectiveness, efficiency, and satisfaction with which a specified set of users achieve a specified set of tasks in a particular environment. The interface may incorporate all the functions specified in the requirements. However, if the user perceives that the interface is not usable, it may not be accepted.

Jakob Neilsen in his book**,** *Usability Engineering*, talks about the importance of creating software that meets certain usability goals and objectives. You can't tell if the software has met usability goals and objectives if you haven't clearly defined them before the product is designed. Usability goals and objectives should include one or more of the factors listed in the table titled

Usability Factors.

Usability Factors

Factor	Description
Usefulness	• Degree to which an interface enables a user to achieve specific goals • Assessment of the user's motivation for using the interface
Effectiveness (ease of use)	• How successfully does the interface allow users to perform their work? • Usually defined quantitatively • Usually tied to some percentage of total users
Learnability	• User's ability to operate the system to some defined level of competence after some predetermined amount of training Ability of infrequent users to relearn the system after periods of inactivity
User Satisfaction	• User's perceptions, feelings, and opinions of the interface • Users are more likely to perform well on a system that meets their needs and provides satisfaction than one that does not

Usability *goals* are not measurable. They should be broken down into usability *objectives* which are more specific and detailed, and most importantly, are *measurable*.

For example, stating a usability objective concerning learnability might be stated as - "*less training will be needed to prepare the user for the new product than was needed to use the current product."* The measurable objective that goes along with this objective could be stated as - "*after 4 hours of training, 90% of the users will be able to data entry of a case report form within 5 minutes."*

Once goals and objectives are defined, usability testing techniques can occur. There are two main types of testing techniques*:* heuristic evaluation and user testing. One type of testing involves usability experts and developers , while the other involves users.

Heuristic evaluation involves having a small set of evaluators examine the interface and judge its compliance with recognized usability principles (the "heuristics"). The table titled, *Ten Usability Heuristics*, lists the ten principles that Jakob Neilsen says should be part of any heuristic evaluation. The important thing to remember is that the heuristic evaluation does not involve users. Heuristic evaluations can eliminate a number of usability problems without wasting user's time, who sometimes can be difficult to find and schedule in large numbers.

Ten Usability Heuristics

	Visibility of system status
	System matches real world
HEURISTIC	User control and freedom
PRINCIPLES	Consistency and Standards
	Error Prevention
	Recognition rather than recall
	Flexibility and efficiency of use
	Aesthetic and minimalist design
	Help users recover from errors
	Provide help and documentation

The other type of usability test is the *user test*. Getting potential users involved in this hands-on testing activity is crucial early on in the design process. This type of usability testing directly involves the end user and can be conducted several ways, the two most common being verbal protocol and questionnaires. Usability testing using verbal protocol involves observing a number of people who represent the end user while they use the product. The users speak their thoughts out loud as they use the product (verbal protocol). The observers note where the users encounter difficulties or confusion. Observation is followed by discussion between the observers and the users, to gather additional information. The usability team then analyzes the data and produces a report summarizing the data and recommendations.

There are many books which are available that lay out who should be involved in the tests and how the tests should be conducted. A very good resource is Jeffrey Rubin's book, *Handbook of Usability Testing*.

Some strong words of advice from usability experts:

- **Don’t let developers test their own products**. Testers need to bring in a fresh view from outside the developers environment. Usability experts and human factors professionals who are not biased in any way make the best test participants. If you don't have these experts at your disposal you might want to try users, graphic designers, and managers. The key is to get the appropriate change in perspective.
- **Don't save usability testing for the end -** UID is an iterative process. In older methodologies usability testing was not considered until the end of the development cycle. While this is better than not considering usability at all, there are major benefits in considering usability at each of the key phases of the development cycle. These benefits include:
  – Improved user acceptance through feedback on early design decisions
  – Decreased cost of modification
  – Improved market acceptance and lower risk

See the appendix for other useful web sites and publications dealing with usability engineering.

Iterative Design Really Works

The iterative design process may look like it takes more time. In fact, if it is done correctly, it will take less time. The work done in early iterations of the design phase can build interfaces that take less time to implement in the later iterations.

Successful products have interfaces that are useful, learnable, and effective. This requires a continual iterative effort that involves the user throughout the process. Remember two things:

"Know thy users", and "Iterate, iterate, iterate."